Detector de Casas Municipales en Imágenes Aéreas (DCMIA)

Documento de especificación de requisitos del sistema (SyRS)

14 de marzo de 2024

Hugo Gil Parente y Javier Martínez Madruga

https://github.com/jamarma/AIVA_2024_DCMIA

Índice

1. Introducción

En esta sección se describe el propósito del sistema, el alcance que obtendrá, se definen los términos técnicos usados en el documento y se incluye un resumen general del sistema.

1.1. Propósito

El propósito fundamental del sistema es proporcionar una solución robusta y fácil de usar para la detección automática de casas y sus respectivas coordenadas en imágenes aéreas de un municipio.

1.2. Alcance

El sistema desarrollado, con nombre *Detector de Casas Municipales en Imágenes Aéreas* (DCMIA), permitirá la detección y localización automática de casas en el municipio del cliente a partir de imágenes aéreas. El usuario final tendrá acceso a un servidor de procesamiento mediante una interfaz gráfica donde, después de cargar una imagen aérea del municipio, recibirá información sobre el número de casas detectadas en la imagen y las coordenadas de los rectángulos que delimitan cada casa.

El objetivo principal es la reducción de costos asociados a la inspección manual de las imágenes, así cómo el procesado de más cantidad de datos en menos tiempo.

1.3. Definiciones

Definiciones de los términos técnicos utilizados en el documento:

- ***Bounding box.*** En el contexto de visión artificial y detección de objetos en imágenes, es un rectángulo que rodea o encierra un objeto en una imagen y se utiliza para definir su ubicación y tamaño en la imagen.
- ***Ground truth.*** Conjunto de datos de referencia que contienen la ubicación y extensión exacta de las viviendas en las imágenes aéreas.
- ***Red neuronal.*** Modelo matemático inspirado en el funcionamiento del cerebro humano, que consta de un conjunto de 'neuronas' interconectadas que procesan y transmiten señales entre sí.
- ***Deep learning.*** Técnica de aprendizaje automático que utiliza redes neuronales profundas para aprender automáticamente tareas a partir de datos.

- ***Transfer learning.*** Enfoque de *deep learning* que utiliza un modelo entrenado para una tarea como punto de partida para otro modelo que realiza una tarea similar
- ***Open source.*** Sistema o software cuyo código fuente es accesible y puede ser modificado, mejorado y redistribuido por cualquier persona.
- ***Multiplataforma.*** Capacidad de un software o sistema para funcionar en diferentes plataformas de hardware o software sin necesidad de modificaciones significativas.
- ***Framework.*** Conjunto estructurado de conceptos, prácticas y herramientas que proporciona una base para el desarrollo y la implementación eficientes de software.

1.4. Resumen

El sistema DCMIA se desarrollará para satisfacer las necesidades de un ayuntamiento que busca conocer las casas que tiene su municipio a partir de imágenes aéreas. Sus funciones serán detectar casas mediante algoritmos de visión artificial y proporcionar las coordenadas de localización de las casas en las imágenes, realizándose todo el procesamiento en un servidor en la nube. Los usuarios del sistema serán personal del ayuntamiento no expertos en visión artificial, por lo tanto, se proporcionará una aplicación gráfica para controlar el sistema de forma fácil e intuitiva.

2. Referencias

[1] Models and pre-trained weights; Torchvision 0.17 documentation. `https://pytorch.org/vision/stable/models.html#object-detection-instance-segmentation-and-person-keypoint-detection`. [Accessed 09-03-2024].

[2] NumPy. `https://numpy.org/`. [Accessed 09-03-2024].

[3] Opencv. `https://opencv.org/`. [Accessed 09-03-2024].

[4] PyTorch. `https://pytorch.org/`. [Accessed 09-03-2024].

[5] Welcome to Flask. `https://flask.palletsprojects.com/en/3.0.x/`. [Accessed 09-03-2024].

[6] Welcome to Python. `https://www.python.org/`. [Accessed 09-03-2024].

[7] ¿Qué es SQL? - Explicación de lenguaje de consulta estructurado (SQL). `https://aws.amazon.com/es/what-is/sql/`. [Accessed 09-03-2024].

[8] Jacob Thornton Mark Otto and Bootstrap contributors. Bootstrap. `https://getbootstrap.com/`. [Accessed 09-03-2024].

[9] Ultralytics. Yolo. `https://docs.ultralytics.com/es`. [Accessed 09-03-2024].

[10] Youzi Xiao, Zhiqiang Tian, Jiachen Yu, Yinshu Zhang, Shuai Liu, Shaoyi Du, and Xuguang Lan. A review of object detection based on deep learning. *Multimedia Tools and Applications*, 79(33–34):23729–23791, June 2020.

3. Requisitos

En esta sección se enumeran los requisitos del sistema.

3.1. Requisitos funcionales

- **Detectar casas**. Se deberán identificar casas en imágenes mediante la detección de sus *bounding box*.
- **Proporcionar número de casas y coordenadas.** El sistema deberá devolver el número de casas detectadas y las coordenadas píxel del *bounding box* de cada casa.
- **Procesamiento en un servidor.** El usuario del sistema deberá poder subir las imágenes a un servidor que realice el procesamiento y le devuelva los resultados.

3.2. Requisitos de usabilidad

El sistema deberá contar con una interfaz gráfica que permita a los usuarios no expertos usar el sistema de forma fácil e intuitiva. Deberán poder cargar imágenes y visualizar los resultados de la detección de manera clara. La interfaz gráfica deberá ser una aplicación web para asegurarnos la compatibilidad multiplataforma y evitar que el usuario tenga que instalar ningún software.

3.3. Requisitos de rendimiento

- **Precisión alta.** El sistema de detección debe manejar un error inferior al 10 %. Esta métrica de acierto se calculará a partir del *ground truth* proporcionado por el cliente.
- **No tiempo real.** No es necesario que el sistema de detección se ejecute en tiempo real.

3.4. Requisitos de interfaces

Los usuarios registrados en el sistema se almacenarán en una base de datos dentro del servidor.

3.5. Requisitos de seguridad

La interfaz gráfica deberá contar con un sistema de login que brinde acceso privado a los usuarios al sistema de detección y su servidor de procesamiento asociado.

4. Verificación

En esta sección se verifica que los requisitos de la Sección 3 están bien formados y se estudia la viabilidad de cada uno de ellos.

4.1. Requisitos funcionales

- **Detectar casas.** La detección de casas en imágenes pertenece a un problema bastante estudiado en el estado del arte de la visión artificial: la detección de objetos [10]. Existen algoritmos y modelos de *deep learning* pre-entrenados [1, 9] que, usando *transfer learning* con datos propios, pueden llegar a ofrecer resultados satisfactorios. Esta tarea se llevará a cabo usando el lenguaje de programación Python [6] y el *framework* PyTorch [4]. Sin embargo, detectar tantos objetos pequeños (casas) en imágenes de gran resolución, puede no ser una tarea trivial.

  El *ground truth* proporcionado por el cliente para realizar el entrenamiento consta de 16 imágenes binarias (*.tif*) con una resolución de 5000x5000 píxeles donde los píxeles con valor 255 pertenecen a las casas y los píxeles con valor 0 al fondo. Es posible que sea suficiente para obtener buenos resultados.

- **Proporcionar las coordenadas.** Obtener las coordenadas píxel de los *bounding boxes* de las casas detectadas es una tarea directa y se puede realizar utilizando bibliotecas *open source* como OpenCV [3] y NumPy [2].

- **Procesamiento en un servidor.** Manejar las comunicaciones entre una aplicación web cliente y un servidor es una tarea factible usando *frameworks* como Flask [5]. Además, la posibilidad de utilizar este tipo de *frameworks* basados en Python permite una fácil integración del algoritmo de detección de casas en el servidor y, de esta manera, realizar todo el procesamiento dentro del mismo.

4.2. Requisitos de usabilidad

La aplicación web para usar el sistema se desarrollará utilizando el *framework* Flask basado en Python y la biblioteca de diseño web Bootstrap [8] que permite crear interfaces gráficas profesionales de manera sencilla. De esta manera se conseguirá una interfaz intuitiva y fácil de usar por un usuario no experto. Las aplicaciones web no necesitan instalación, se pueden ejecutar en cualquier navegador como Chrome, Edge o Firefox.

4.3. Requisitos de rendimiento

- **Precisión alta.** La detección de objetos pequeños (casas) en imágenes de alta resolución no es una tarea trivial y el error final del sistema vendrá determinado por la precisión de los algoritmos de detección de objetos del estado del arte y la calidad y cantidad de datos proporcionados como *ground truth*.
- **No tiempo real.** El tiempo de procesamiento de cada imagen vendrá determinado por la capacidad de cómputo del servidor contratado y el coste computacional del algoritmo de detección de objetos usado.

4.4. Requisitos de interfaces

Usar una base de datos SQL [7] dentro de un servidor es una tarea común y se puede integrar con la aplicación web fácilmente a través de Flask.

4.5. Requisitos de seguridad

La creación de un sistema de inicio de sesión es una tarea común en el desarrollo de aplicaciones web, Flask proporciona herramientas para su implementación.